AMENDMENTS TO THE SPECIFICATION:

Please amend the paragraph beginning at page 6, line 19 as follows:

Referring to Figure 1, shown is an example of an embodiment of a computer system. The computer system 10 includes a computer processor 20 connected to a data storage device 22 by communication means 24. In the computer system 10, the computer processor 20 may be any one of a variety of commercially available processors, such as an ~~INTEL based~~ INTEL ® processor. It should be noted that other computer processors may be used in an embodiment and neither the type nor number of processors should be construed as a limitation.

Please amend the paragraph beginning at page 8, line 11 as follows:

The monitor process 30 may be a machine executable program written in any one of a variety of commercially available programming languages, such as C++. An example of a monitor process is included in the commercially available implementation of ~~NUMEGA BOUNDS CHECKER~~ NUMEGA™ BOUNDSCHECKER™. Generally, the monitor process 30 gathers information and data in accordance with the execution of the program under test 44 which in this example may be referred to as test.exe. The machine executable program 44 being monitored in this particular example may be written in one of a variety of conventional, commercially available languages, such as C, and C++, for which instrumentation may be provided. One technique of how instrumentation of a program may be performed is described in U.S. Patent Application No. 08/916,125, now issued U.S. Patent No. 5,987,249, entitled "IR Code Instrumentation".

Please amend the paragraph beginning at page 9, line 9 as follows:

It should be noted that the monitor proxy DLL 42 may be included automatically at link time into the machine executable 44 in this particular embodiment as a result of code instrumentation. Alternatively, other embodiments may employ other techniques to cause DLL42 to be loaded into process execution space, for example, as described in "Programming Applications for MICROSOFT® WINDOWS®", 4th Edition, by J. Richter, Microsoft Press, September 1999. In this particular embodiment, the monitor proxy DLL 42 is included in the commercially available product NUMEGA™ TRUE TIME™ in conjunction with the monitor process 30, for example, using the techniques set forth in U.S. Patent Application No. 08/916,125, now issued U. S. Patent No. 5,987,249, entitled "IR Code Instrumentation". Generally, the monitor proxy DLL 42 includes monitoring routines which report information back to the monitor process 30. In this particular embodiment, the monitor proxy DLL 42 includes code that monitors events related to the loading of various types of libraries. For example, the libraries monitored by routines in the monitor proxy DLL 42 include shared libraries or DLLs as well as OCX or ~~ActiveX~~ ACTIVEX ® Control libraries. Generally, as known to those skilled in the art, DLLs and OCXs include shareable machine executable code that may be used for example by program 44. The proxy monitor DLL 42 includes routines that send load event information to the monitor program 30 using connection 32.

Please amend the paragraph beginning at page 10, line 3 as follows:

In this embodiment, the monitor proxy DLL 42 may receive DLL load information by scanning the process virtual address space, or, by monitoring calls to operating system routines that load the DLLs, such as the LoadLibrary routine from MICROSOFT® WINDOWS®. Information may be transferred to the monitor 30 using shared memory as the communication channel 32. It should be noted that other embodiments may employ other techniques in accordance with each implementation.

Please amend the paragraph beginning at page 10, line 9 as follows:

The library load information communicated from the monitor proxy DLL 42 to the monitor process 30 may be achieved, for example, in a WINDOWS® programming environment, such as using WINDOWS® 95 or WINDOWS® NT. In this environment, programming applications may be invoked from a menu option. For example, using a "start" option selected from a menu display, the machine executable program 44 may be selected to begin execution. Once the program 44 to be tested or monitored is executed, the monitor process 30 is invoked indirectly from code included in the monitor proxy DLL. The monitor process 30 is invoked indirectly because previously the machine executable program 44 was linked for instrumentation and monitoring.

Please amend the paragraph beginning at page 10, line 18 as follows:

The monitor proxy DLL 42 monitors operating system events for loading machine executable code or libraries such as DLL1 36 and DLLN 40. The monitor proxy DLL 42 includes code to accomplish this, for example, by monitoring user API calls made within the user supplied portion 34. For example, the monitor proxy DLL may monitor specific user APIs such as CoCreateInstance and Load Library which are APIs used in the WINDOWS ® environment for loading DLLs. It should be noted that other embodiments may include monitoring of other particular APIs from within user code 34. Additionally, other techniques may be used to gather information regarding software components used with a program being tested 44.

Please amend the paragraph beginning at page 11, line 20 as follows:

The alternate embodiment as depicted in Figure 3 may occur, for example, by invoking the monitor process directly from the "start" program menu as in a WINDOWS® 95 or other similar programming environment. Subsequently, from within the monitor process 30, a command or graphical user interface may be displayed and the user may enter commands to invoke the machine executable program to be tested 50.

Please amend the paragraph beginning at page 12, line 19 as follows:

It should be noted that the embodiments previously described with regard to Figures 2 and 3 are two of a variety of different techniques that may be used to gather various software component information. In this particular embodiment the software components for which information is being gathered are libraries, such as DLLs or OCXs as known in the WINDOWS

® environments. Other types of programming environments may include different types of software components other than a library such as a DLL or OCX.

Please amend the paragraph beginning at page 13, line 9 as follows:

In the previously described embodiment of Figure 2, the monitor proxy DLL 42 is included as a portion of the program being tested 44 to monitor for specific calls and communicate information to the monitor process 30 when a particular software component is loaded. This DLL 42 is automatically included in the linked program 44 as a result of instrumentation of the program. In this embodiment, for example, DLL 42 may be included in a commercially available product such as NUMEGA™ TRUE TIME™. The technique of Figure 2 may be used with a program that uses code instrumentation. For example, the program 44 may be an instrumented machine executable program produced using C source code.

Please amend the paragraph beginning at page 20, line 16 as follows:

Referring to Figure 8B, shown is an example of an embodiment of a session object. The session object 72 in this example includes a user field 72a, a date/time field 72b, a comment field 72c, a pointer to the system configuration 72d, a pointer to the project 72e, an identifier as to the product submitting the session 72f, a list of components included in this session 72g, and other information as indicated by field 72h. The user field 72a identifies a particular user-identifier, such as a login identifier of a user. This may be represented, for example, as an alpha-numeric string. The date/time field 72b is a date/time stamp information as to when the session occurred.

This may be stored, for example, as an alpha-numeric string. The comments 72c may be a text field describing the particular session, for example, "test run for bug correction number nnn". Field 72d is a pointer to the associated system configuration object, such as indicated by arrow 75f of Figure 5. Field 72e is a pointer to the project object, such as indicated by arrow 75e of Figure 5. Field 72f is an indicator as to the product submitting the session. In this embodiment, this field indicates which monitor product in the commercially available product suite, such as NUMEGA™ DEVPARTNER STUDIO™, the identified user 72a was running. Field 72g is a list of modules associated with this session. For example, this list of associated modules may be represented as arrows 77a, 77b, and 77c of Figure 5. Field 72h may include other information related to the session being represented in a particular embodiment.

Please amend the paragraph beginning at page 21, line 10 as follows:

Referring to Figure 8C, shown is an example of an embodiment of a bug report object that may be included in the previously described representation of the database schema 71. The bug report object 300 in this embodiment includes a pointer to a system configuration object 300a, a list of modules 300b, a pointer to a project object 300c, a pointer to a session file 300d, bug report identifying information 300e, and other bug report data 300f. This object may be created, for example, in conjunction with a bug report submitted to the e-mail reader tool 62, and included in the database. It should be noted that the fields 300a-300c refer to objects previously described. Data associated with this object may be retrieved and used, for example, as will be described in paragraphs that follow, when querying the database regarding a version of a software module associated with a bug report. The pointer to a session file 300d identifies a

corresponding data file related to the bug report, for example, a data file related to test coverage data as may be generated by a testing tool such as NUMEGA™ TRUE TIME™ or NUMEGA™ TRUE COVERAGE™. The bug report identifying information 300e may be one or more fields of data identifying the reported bug. For example, this information 300e may include data such as a priority, who entered the bug report, the date of bug report submission, a description of the problem, product information associated with the reported bug, and the like. It should be noted that, as with other data representations described herein, the fields included may vary in accordance with each implementation.

Please amend the paragraph beginning at page 22, line 21 as follows:

At step 98, system configuration information is gathered. It should be noted that in this particular embodiment the system configuration information includes data as represented and stored in the system configuration object in accordance with the data representation 71. The system configuration information generally includes data which uniquely identifies a particular system configuration. For example, the system configuration information gathered in step 98 may include a portion of the information as may correspond to, for example, WINDOWS ® registry settings.